社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

特定分野のソフトウェアに関する特性の相関を用いた要求獲得法

長田　晃† 小澤　大伍† 海谷　治彦† 海尻　賢二†

† 信州大学 大学院　〒 380-8553 長野市 若里 4-17-1
E-mail: †{csada,kaiya,kaijiri}@cs.shinshu-u.ac.jp, ††t05a516@amail.shinshu-u.ac.jp

あらまし　ソフトウェア要求を仕様化する際にはドメイン固有の特性と特性間の相互関係を考慮しなければならない。しかし要求獲得時には、顧客のニーズを理解することに専念し、それらを考慮するのを忘れることがある。本論文では、ドメイン特性とその相関をあらわすメタモデル、および既存システムのドキュメントを用いて当該ドメインのモデルを構築する手法について提案する。特性の順序付けにより、それぞれの特性の相関も見つけることができる。このモデルを使うことで、アナリストはドメイン特性をチェックすることができるため、完全で、曖昧性のない要求仕様書を作ることができる。また、要求獲得時および変更時に、要求間の相関を考慮することもできるため、仕様書の正当性と無矛盾性を高めることができる。ケーススタディを通してこのモデルと方法論の有効性を確認する。
キーワード　要求工学，要求獲得，ドメインモデリング.

Modeling Software Characteristics and Their Correlations in A Specific Domain by Comparing Existing Similar Systems

Akira OSADA†, Daigo OZAWA†, Haruhiko KAIYA†, and Kenji KAIJIRI†

† Graduate School of Science and Technology, Shinshu University　4-17-1, Wakasato, Nagano city, 380-8553, Japan
E-mail: †{csada,kaiya,kaijiri}@cs.shinshu-u.ac.jp, ††t05a516@amail.shinshu-u.ac.jp

Abstract　Software in a specific domain has several characteristics and each characteristic should be fixed when the software requirements are specified. In addition, these characteristics sometimes correlate with each other. However, we sometimes forget to specify several characteristics and/or to take their correlations into account during requirements elicitation. In this paper, we propose a meta-model for representing such characteristics and their correlations, and also propose a method to build a model for a specific domain by using documents about existing software systems. By using our model for a domain, a requirements specification for a system in the domain could be complete and unambiguous because requirements analysts can check the characteristics that should be decided. The specification could be also correct and consistent because the analysts can know side effects of a requirement change by using correlation among the characteristics. We have applied our methods to a case study for confirming the usefulness of such model and the methods.
Key words　Requirements Engineering, Requirements elicitation, Domain Modeling.

1. はじめに

我々は、既存システムのリソースを使うことで、新しいシステムの要求を抽出・定義するのに役立つドメイン知識をひきだすことができると考えた。例えば、要求アナリストが図書館のユーザ登録システムに関する要件を引き出すときは、暗黙的であれ、明示的であれ、似たようなシステムやそのドキュメントを参考にするだろう。本論文では、そのような知識を管理するモデル、モデルの構築の方法、および、そのような知識の使い方を提案する。本モデルと方法論の期待される結果は以下である。

- ドメインで決定すべき特性を仕様化し忘れることを避けることができる
- 相互に矛盾する特性を仕様に含むことを回避できる
- 特性の不正確な実現を避けることができる

このモデルと方法論の主要なアイデアは、ドメイン知識とドキュメントから、機能を順序付けることである。図 1 に三つの類似システムを比較し、順序付けする例を示す。それぞれのシステムには、“ユーザー数” と “パスワード変更” という二つの機能を持つとする。図では、それぞれのシステムが持つ “ユーザー数” に “F”, “G”, “H” とラベルを付けた。ここで、要求仕様としては “最大ユーザー数” という特性を見つけることができる。3 つの機能は数の観点から順序付けでき、その順序付けから好悪の評価基準を推定することができる。ここで言うと、

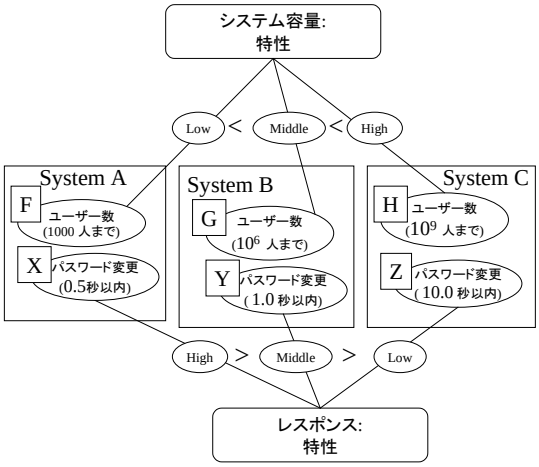

図 1　類似システムの比較と順序付けの例

“ユーザー数は多い方が良い” という評価基準である。また、このような比較により、特性が必須か付属かを認識するのにも役に立つ。例えば、パスワード変更という特性で、もし、3 つの機能全てが一定の値（例えば 0.5 秒以内）という属性を持つならばならば、その値は必須であろう。逆に、図 1 のようにそれぞれの機能が違う値を持つとき、特性は必須ではなく、変動のある特性であると考えることができる。

図 1 に示すようなモデルを利用して、特性の無矛盾性と正当性を確認する方法論を提案する。顧客のリクエストにより、ある特性が変更されたとき、暗黙的に関連してる他の特性についても再考慮しなければならない。もしそれらの特性について再考慮しないとしたら、顧客のニーズを満たさないシステムになるからである。それはすなわち、相互に関連している特性を知っていなければならない、ということを意味している。我々は、機能と属性を順序付けるデータを用いてそれぞれの相関を系統的に計算することで、そのような特性を見つけることができると考えている。図 1 は相関の簡単な例である。ユーザー登録という機能について、“H”、“G”、“F” という順序に着目し、“登録数は多い方が良い” という評価基準を発見できる。一方、“パスワード変更” について、“X”、“Y”、“Z” という順序があるとすれば、“レスポンスが速いほうが良い” という基準が発見できる。すると、2 つの機能の順序は正反対であるため、2 つの機能の間には逆相関があると推測できる。このような場合、“ユーザー登録” に対して、要求が変更されたときに、相関がついている “パスワード変更” について、再考慮する（しない）、ということを確認することができる。もし再考慮しないとしたら、“パスワード変更” という特性は顧客のニーズ（例えば、レスポンスが良い）に反して決定されてしまい、“パスワード変更” に対する正当性違反が回避できないことになる。

本論文は以下の構成である。2 節では、図 1 のような既存システムについてのメタモデルの説明をする。3 節では、モデルの使い方とモデルの利点について説明する。4 節では、モデルの開発の仕方を表す。5 節では、モデルと方法論の効果性を確認するケーススタディを紹介する。最後に、関連研究と結果のまとめ、今後の予定を述べる。

2.　ソフトウェア特性とその相関のモデル

この節では、初めにメタモデル（ドメインの知識のためのモデルのデータ構造）を紹介する。また、モデルと特性、それらの相関を表現することについての例を示す。

2.1　メタモデル

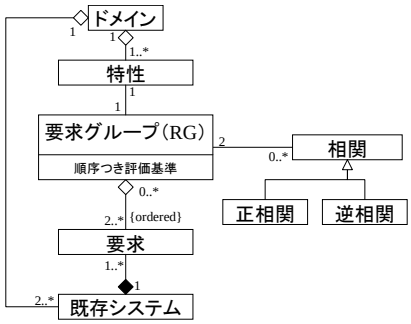

図 2　メタモデル

図 2 は UML により書かれたクラス図で、ドメインを表現するためのメタモデルである。メタモデルは以下のものである。

- アプリケーションには、既存システムのインスタンスが 2 つ以上あるべきである。
- 各システムは文章やユースケースで記述された要求記述から構成される。
- 要求グループは要求の順序付き集合と、その順序付けの仕方から得た評価基準を持つ。集合の要素は 2 以上であるべきである。
- 要求グループは特性と対応している。これは特性が、具体的な要求の順序集合によって仕様化されることを意味している。
- ドメインは、仕様化すべき、または考慮するべきいくつかの特性を持つ。2 つの要求グループは、正相関または逆相関を持つ場合がある。

このメタモデルを使うことで、仕様化するべき特性について考慮するときに、要求の具体的な記述を簡単に呼び出すことができる。さらに、要求の比較によって特性を理解することもできる。なお、特性のインスタンスの名前は、信頼性や使用性などの抽象的な特性を使わなくとも良い。また、現在、間隔尺度、比率尺度については考慮せず、順序尺度にのみ注目している。それは汎用的に要求を比較するときに、間隔や比率を導入するのは困難と思われるからである。例えば、2 つの要求を比較するときに、「他方より 2 倍良い」ことを一般的に述べることは困難と思われる。要求を順序付けるための技術は 4.2. 節で触れる。

2.2　モデル例

図 3 では、相関の表現の仕方と、典型的なモデルの例を示す。簡単のため、既存のシステムを表現するためにユースケース図を使う。また、相関の補足説明のため二次元グラフを付記する。図 3 は図 1 に “逆相関” と “要求グループ” の 2 つのインスタンスを加えた拡張であり、図 3 のアプリケーションドメインは図 1 と同じである．図中の上側にある “要求グループ” は “人数は多い方が良い” という評価基準を持つことから、そのグループについての嗜好と重要性を容易に理解できる。また、“特性” と “要求グループ” は対応付いているため、我々は特性の理解が容易である。“相関” の直感的な具体例は、図 3 中の二次元グラフに示されている。この例では、2 つの要求グループがお互いに関連しており、“ユーザー数” に変更があったときには、“パスワード変更” について再考するべきであることを意味する、となる。さらに、図 3 の例を使って、2 つの要求グループが相関

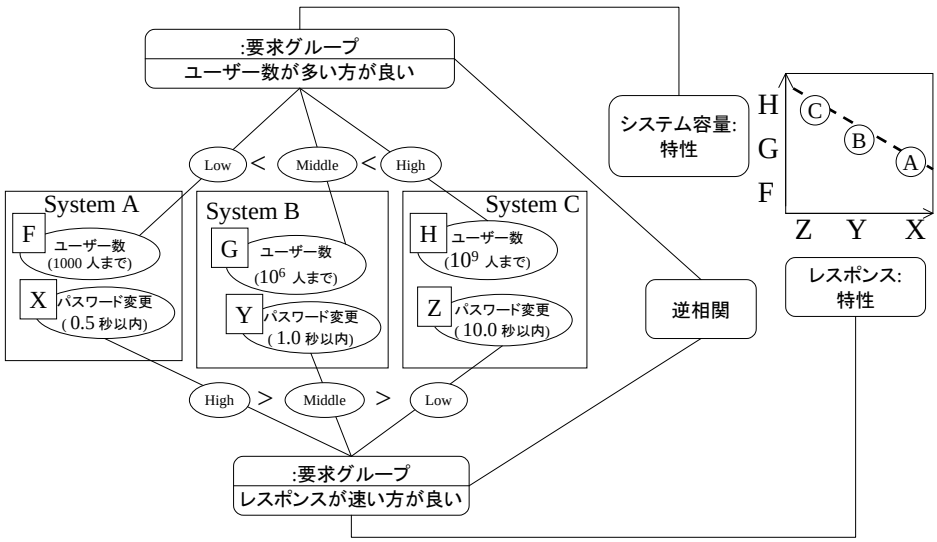

図 3　図 1 を拡張した典型的な例

なのかどうかを決定する仕方の説明をする。この例では、要求 F,G,H は “ユーザー数が多い方が良い” という評価基準により、H > G > F となる。従って、それをグラフの縦軸に並べる。同様に、X、Y、Z を水平な軸に並べる。F と X が System A の部分であるので、グラフの F と X の交差点は System A を意味する。また、他の交差点ポイントは対応する Systems を意味する。グラフにあるように、2 つの順序集合 {F, G, H} と {X, Y, Z} は関連があるように見える。よって、順序集合で表せられる要求グループにも相関があると思われる、となる。なお、間隔・比率尺度ではなく、順序尺度に基づいているため、厳密に相関係数を計算してのテストはしていない。

3.　モデルの使い方

3.1　獲　得　法

モデルに含まれている特性の集合は、仕様化するべきもしくは注意の必要なものであり、少なくとも要求獲得時には話し合わなければならないような特性である。ステークホルダーは自身の願望や嗜好を明示的に把握していない場合があるが、このモデルの評価基準によってそれらを理解・把握することができる。さらに、ステークホルダーは、各々の特性に関連付いた具体的な要求を参照することで、それぞれの特性に関する傾向を把握できる。これにより，ある特性が固定的か変動的か、必須か否かを判断する助けとなる。

3.2　変更要求への対処

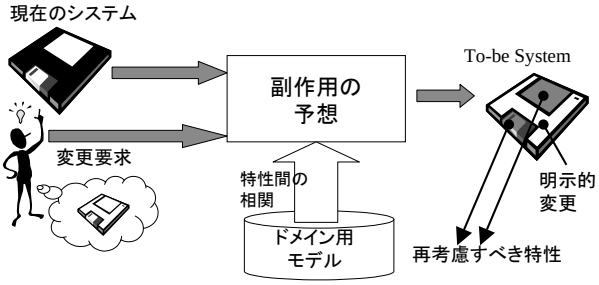

図 4　相関を使った副作用の予想

図 4 で、ステークホルダーからの要求変更への対処を比喩を用いて説明する。図ではディスクの本体の色が、黒から白に変更するという要求変更があったとしている。すると、我々は要求変更の副作用を調査しなければならない。図では副作用とは、ラベルの色とシャッターの色の変更である。もしこれらの副作用に留意しなければ、2 つの問題が起きると考えられる。1 つは不適当という問題である。論理、物理、政治的、法律的または組織的制約など、幾つかの理由で、1 つの変更は別の変更を必要とすることがある。例えば、図 1 では、1000 から 10^9 へユーザー数が変更されたら、パスワード変更のためのレスポンス時間が大幅に衰えることになる。もしアナリストとステークホルダーがこの問題について見過ごすなら、出来上がったシステムは、レスポンスについて顧客のニーズを満たすものとはならないだろう。我々は資金を投入することや特別な技術を導入して、このような問題を回避することができるが、そのような状況を認識し、対処法を決定するのは、要求分析段階であることが望ましい。我々のモデルと方法は、そのような副作用の把握に寄与する。一方、もう一つの問題は、矛盾である。もし 2 つの矛盾した要求、または、両立しない要求があれば、システムは実現しないか、多大な労力とコストが掛かることになる。それゆえに、要求分析段階で、そのような要求の 1 つを削除するか、妥協案または次善案を見つけなければならない。

以上のことから、類似した特性をもつ既存のシステムから計算された相関を使えば、矛盾や不適当の問題を発見するのに役立つ。

3.3　帰納法による変更対処

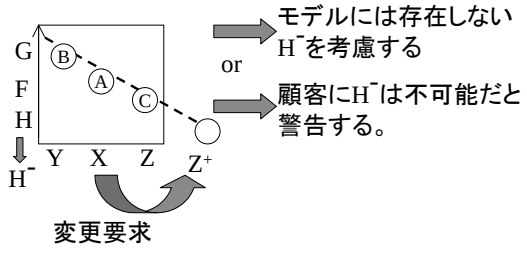

図 5　相関による帰納法

ここで、モデルの最も斬新な利用法を紹介する。一般的に、

モデル内の既存の要求と、特性についての要求は直接的には一致しない。例えば、ユーザー数が 3000, 30 または 10^{30} と要求されたとき、図 1 のモデルの中で直接一致する要求はないため、これらの要求からすぐに副作用を予測することはできない、ということになる。そのため、特性についての帰納法を導入する事とする。図 5 に帰納法の例を示す。図は、副作用を帰納法的に発見する事と、副作用の検討を顧客に要請することの例である。

図 5 に対して、Z^+ という要求変更があったとする。すると線形回帰手法を用いて、図 5 に、H^- という要求が推測される。もし H^- が他の要素との関係上、実現不可能であるとすれば、我々は顧客に Z^+ は重大な副作用を引き起こすことを警告しなければならない。H^- が可能であるとしたら、ステークホルダーに H^- の副作用があることを示し、変更するかどうかをきめさせる。なお、実際は順序付けに間隔尺度を用いないので、線形回帰手法を用いることは不適切である。しかし、我々のゴールは可能性と副作用の傾向を発見することであって、副作用を明確に誘導することではないので、この方法をあえて採用した。

4. モデルの作り方

効果的な要求獲得が可能か否かは、モデルが適切か否かに大きく依存する。モデルが適切であるならば、例えば、既存システムの数が増加しても容易にモデルを保守でき、新知識を既存のモデルに簡単に付加できることになる。ここでは、モデルの作成方法を説明し、モデルが適切であることを示す。

4.1 手　　順

基本的に、モデルは以下の段階で作成される。

（1） あるドメインに属する既存システムについてのリソースを集める。なお、リソースとは、マニュアルやヘルプファイル、または、そのシステム自身である。

（2） それぞれのシステムの属性や機能としての要求を見つける。ここでは、ユースケース図を描くことが役にたつ。すでに分析済のシステムで識別された属性や機能を参考に、新しい属性や機能を識別してもよい。

（3） 要求の集合と、要求グループのインスタンスを作る。(2) で述べたように、前もって把握されている機能や属性を参考に新たな機能や属性を識別するため、類似した要求は既にグループ化されている場合が多い。

（4） それらの要求の間に、評価基準を見つける。たいていは、“何々が好ましい” という表現を使う。(2) で、類似した要求と、ある要求を比較しているので、その際に、一般的な評価基準を既に使っているといえる。

（5） 特性とそれに関連付いた要求グループのインスタンスを作る。使用性とか、信頼性とか、特性を認識しやすくするために、特性のインスタンスに名前をつけることもできる。

（6） 見つけた評価基準から、要求を順序付けする。

（7） 相互の要求集合に順序付けを比較して、要求グループのインスタンスの相関を計算する。

4.2 比較と順序付けの技術

要求間の比較は重要な役割を持つ。ここでは、比較のために以下の技術を用いる。

- 要求に書かれた数字: それぞれの要求が数値的な属性をもつとき、他の類似した要求と、数値で比較することができる。典型的な例は、図 3 の “ユーザー数” や “反応時間” である。
- ユースケース図の構造的周囲 [1]: 既存システムのユースケース図が使えるとき、要求の構造的な類似性を用いることができる。一般的に、類似したユースケース群は，それぞれのユースケースに関連を持つユースケース数に基づいて、順序付けを行うことができる.
- 仕様マッチング [2]: それぞれの要求を機能と見立てれば、要求は事前事後条件を持つ。そこで比較のために、事前事後条件に沿って論理包含を使うことができる。図 3 の “ユーザー登録” を例にするば、G の事前条件は F の事前条件を含み、H の事前条件は G の事前条件を含む。つまり、要求の F,G,H には論理包含により、$H > G > F$ という順序が付くことになる。
- ユースケースポイント [3]: いくつかの機能に対して使用性に注目する場合、振る舞いや相互運用性の側面が重要になる。ユースケースポイントは、システムと外部とのインタラクションの範囲と数に注目しているため、相互作用の面から要求を比較するときには、ユースケースポイントを使うことができる。
- *AHP* [4]: 特定のユーザーの好みに関して要求を順序付けしたいことがある。AHP によって主観的な順序付けを系統的に行うことが可能である。

無論、比較のためには幾つかの技術を同時に使うこともできるし、他の技術を使うこともできる。経験上、要求の認識と要求間の比較はそれぞれを相互に補完しあうため、既存システムのそれぞれのリソースを比較するときに、要求が効果的に認識できるということである。

4.3 スケーラビリティ

新しい既存システムが既存のモデルに追加されるとき、考えなければならない特性の数が増加するのは明白である。それは、ある機能と類似した機能を比較するとき、既存の属性や機能を評価するための新しい評価基準を見つけることができるということでもある。しかし、相関関係を計算する元のシステムが少数であれば、事実上、意味のない多くの相関を持つ組が存在する。もしそれらのペアが多いなら、特性の副作用を効率的にチェックすることができない。しかし、既存システムが増加することで、偶然一致した相関が減少すると考えられる。次の節でその点について確認する。特性間の相関は系統だてて計算されるため、特性が増えても気にしないでいられるかもしれないし、もし無意味なペアが減少しなかったとしても、それらのペアからの結果は排除できるかもしれない。それらの関連の原因や副作用を、自動的とはいわないが、相互作用で検出するつもりだからである。

5. ケーススタディ

この事例では次の仮定の確認を行う。

- 既存システムの比較によるモデル開発は、容易である
- モデルは、ドメインに対して、変動、付属、必須などの特性を知ることに寄与する。
- 特性の相関関係は、特性の関連性を見つけることに役に立つ。
- 相関の質は、既存システムの数が多くなるとき、改善される。

5.1 ケーススタディのデザイン

図 6 はデータフロー図で書かれたワークフローである。このワークフローで次のそれぞれのプロセスを説明する。

- 同じアプリケーションドメインに所属する 3 つのソフトウェアのドキュメント。それらのドキュメントを D1,D2,D3 と表す。

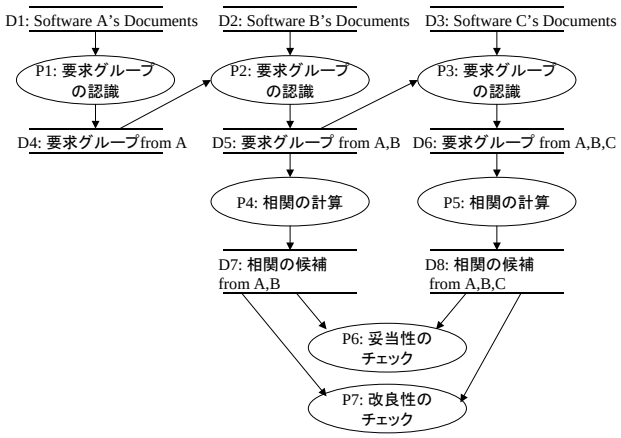

図 6　ケーススタディのやり方 (written in DFD)

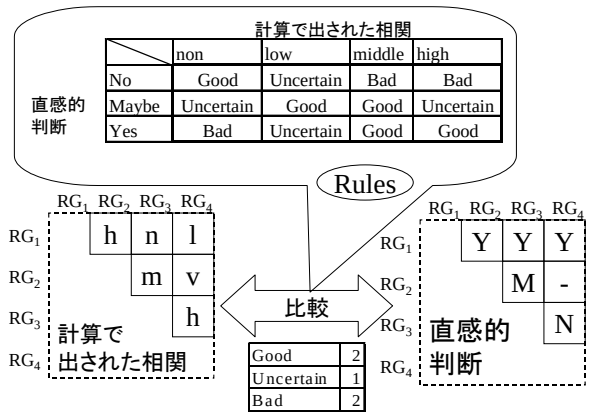

図 7　相関の妥当性の例

- （プロセス　P1）D1 を読み、要求グループと対応する特性および評価基準を認識する。その結果として、分析者は D4 で表される幾つかの要求グループを獲得する。D4 は 1 つのソフトウェアから作られるため、要求グループの要素は 1 つであり、順序づけされない。それゆえ、この時点では相関は考慮されない。
- （プロセス　P2）D2 と D4 から、分析者は上記と同様に分析する。結果として、分析者は、D5 の拡張された要求グループを獲得する。
- （プロセス　P3）D5 と D3 から、上記と同様に分析を行う。結果として、D6 の、より拡張された要求グループを得る。
- （プロセス　P4）D5 を使って、要求グループの全てのペアの相関を計算する。そのような相関のデータを D7 で表す。
- （プロセス　P5）D6 を使って、要求グループの全てのペアの相関を計算する。それらの相関が D8 である。
- （プロセス　P6）D7 と D8 の内容のそれぞれを、図 7 の手法を用いて、計算された相関が直観的に正しいかどうかを調べる。
- （プロセス　P7）D7 と D8 を比較して、分析者は相関の質が改良されたかどうかを調べる。図 8 の説明は後述する

計算された相関が妥当か否かは図 7 に示すように，計算された相関 (h=高 m=中 l=低 v=計算不能 他は相関が無い) が熟練者の直感的な判断にどの程度一致しているかで調べる．P4,P5 の比較から相関の質の向上を図 8 に示す手順で調べる。意味的、因果的な関係がないにもかかわらず、相関付いた特性のペアをノイズペアと呼ぶことにする。もしノイズペアが多ければ、矛盾と不適当を検出するために多くの努力が必要になる。そこで、既存システムの数が増加するとノイズペアが減少するか否かを調べることで、相関の品質改善を調査する。尚，対象とする既存システム数が増加すれば，特性の数が増加する場合があるため (図 8 では 4 から 6 に増加)、ノイズペアの比率の比較は、(1) 従来からあった特性に関して (図中の P と Q)、(2) それぞれの段階での全特性数に基づいて (図中の P と R) の二通りを行う．

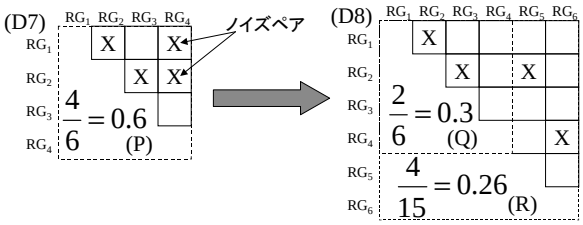

図 8　相関のノイズ削除の例

表 1　要求グループの数: D4, D5, D6

		一致したグループ数
D4 (from D1)	17	17
D5 (from D4 and D2)	25	4
D6 (from D5 and D3)	33	3

5.2　結　　果

この例では、音楽プレーヤーのアプリケーションドメインを選択した。このドメインに詳しい著者の 1 人が、この例で分析者の替わりをした。彼はウェブサイトから 3 つのソフトウェアをダウンロードした。選んだのは windows 上で動くフリーウェアである。ソフトウェアは windows ヘルプファイルやユーザマニュアル（.hlp、.chm）を含み、それらを D1,D2,D3 とした。それぞれに文書のサイズは PDF に換算しておよそ 4, 20, 102 ページほどだった。マニュアルは、アップデートと変更履歴、プレーヤーの使い方の記述があった。そして、それらを理解するために、それぞれのプレーヤーのユースケース図を描いた。

- **結果 1**: D4,D5,D6 についての結果が表 1 であり、要求グループとして認識された数である。表の右端の列は、グループ内の特性がほぼ同一であるために順序がつけられなかったグループの数を示す。
- **結果 2**: 分析者は P1 では機能をリストアップするだけだった。しかし、P2 と P3 においては、既知の要求と比較できるため、要求グループの認識は容易だった。つまり、それぞれのグループの評価基準と特性を知ることが出来たということである。彼は、4.2 節の技術を常に使い、“構造的周囲の把握” という技術を最もよく使った。
- **結果 3**: 1 つのソフトウェアにだけしか存在しない要求（機能）を 7 つ見つけた。図 2 では、要求グループの数は 2 以上で定義すべきであるとなっているため、要求グループと関連付け出来なかった。
- **結果 4**: 結果 1 と表 1 で述べたように，一致した要求グループがいくつかあった．それらの内容を調べると，このドメインにおいて必須な特性を示していることがわかった．例としては，“リスト機能” や “1 回演奏” などである。
- **結果 5**: 一方，グループ内の特性が全くばらばらなものもあった．そのような特性の内容を調べると，本ドメインで典型的な決め方が無く，変動しやすい特性であることがわかった．例えば、“ルックアンドフィール（たいていはスキンと呼ばれ

表 2　相関の妥当性

	D7 (from D1, D2)	D8 (from D1, D2, D3)
グループの数	25	33
相関の組 (α)	300	528
パターン v の組 in Fig. 9 (β)	90	93
$\alpha - \beta \Rightarrow$ Total	210 (100%)	435 (100%)
Good	50 (24%)	137 (32%)
Uncertain	32 (15%)	158 (36%)
Bad	128 (61%)	140 (32%)

表 3　P7 の結果：相関のノイズ減少（図 6 と 8）

	D7 (from D1, D2)	D8 (from D1, D2, D3)	
グループの数	25		33
利用可能な相関の数 (α)	300		528
全体の相関の数 (γ)	210	155	362
利用可能な相関の比率 γ/α	0.70 (P)	0.52 (Q)	0.69 (R)

る）” と “ソフトウェア状態の表示（曲の時間経過）” などである。

- **結果 6**: 計算された相関を持つ要求グループのペアの内容を調査し、ほとんどのペアはお互いの因果関係を持つように思われた。例えば、“ソフトウェアのステータスは詳細表示されるのが良い” というペアと、“動作が軽快（ショートカット）” は相関づいており、それらは因果関係を持っていると思われる特性である。図 7 と同様に相関の妥当性を調べた結果を表 2 に示す．“Good” の割合が 24%から 32%に増加し、“Bad” は 61%から 32%に減少したため、D8 の相関は D7 より改善された。
- **結果 7**: 図 8 のように、分析者は既存システムの数の増加により、相関の質が改良されたということを調べた。結果が表 3 である。統計的検定は行わなかったが、((P) > (Q), (P) > (R)) のように質が改良される傾向がみられた。計算で出されたノイズペアのほとんとが、意味的に、または因果的に、関係がないと思われた。

5.3 議　　論

このケーススタディの結果から、ほとんどの仮定が直感的に確認できた。特に、本論文の骨子である類似した属性および機能の比較は役に立つと思われる。また、4.2 節の要求の順序付けの技術も役に立つ。しかしながら問題もあった。要求は全順序で順序付けするように図 2 のメタモデルで定義されているが、実際には半順序しかつけられないケースがあった。よって、メタモデルの修正をする必要があると思われる。

特性間の相関は、結果 6 にみられるように副作用や因果関係を見つけるために役に立つと思われる、結果 6,7 から、相関は、既存システムの数が増えれば妥当性と質が改良されるため、スケーラブルだと思われる。

6. 関 連 研 究

本稿の研究と最も類似した研究は QFD [5] である。QFD では、ステークホルダーの要求とソフトウェア特性を対応付けることで、要求定義を行う。我々の研究は注目すべきソフトウェア特性を模索することに重点をおいていることが QFD と異なる。

我々の研究はソフトウェアの品質特性を扱っており、実際、ISO の品質特性標準 [6] に関する調査が本研究のヒントの一つとなった。我々の研究は結果として非機能要求 [7] を扱っているが、非機能要求と機能要求を分離して扱っていない。

ドメイン知識に関する研究は要求工学の分野でも数多くみられる。例えば、FODA [8]、LEL [9] 等である。FODA では抽象化によってドメイン特性を識別しているが、我々の研究では比較による識別戦略となっている。LEL は自然言語処理に基づきドメイン知識を容易に獲得することに焦点がおかれているが、我々は要求仕様の妥当性や完全性等を高めるという目的に特化してドメイン知識を収集している。

DDP [10] や WinWin [11] 等の優先度付け技術が要求工学では幅広く認知されているが、これらも我々の手法における “特性比較ツール” の一つとして利用可能だと思われる。

7. ま と め

本稿では、ドメインモデルとモデル構築手法を提案した。このモデルは要求仕様の獲得および変更に際して、要求仕様書をより妥当、より無矛盾、より完全かつより非曖昧にするように設計されている。このモデルは同じドメインに属する既存システムの特徴の比較に基づき構築され、モデルの構築の手間は既存システムの個数に対してスケーラブルである。すなわち、既存システムの個数が増えることで，モデルの妥当性や品質は改善する．ケーススタディを通して、部分的ではあるが我々のモデルと構築手法が有効であることを確認した。本モデルを利用するに際して分析者の主観に頼る部分があるため、ケースツールによる利用のサポートが有用である。今後、本モデルを利用した要求獲得と保守の方法およびツールの設計・実装を進めてゆきたい。

文　　献

[1] Haruhiko Kaiya, Akira Osada, and Kenji Kaijiri. Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems. In *Proc. of RE04* pp. 112–121, Sep. 2004.

[2] Amy Moormann Zaremski and Jeannette M. Wing. Specification Matching of Software Components. *ACM Trans. Software Eng. and Methodology*, Vol. 6, No. 4, pp. 333–369, Oct. 1997.

[3] Geri Schneider and Jason P. Winters. *Applying Use Cases: A Practical Guide*. Addison-Wesley, 2nd edition, 2001.

[4] Thomas L. Saaty. *The analytic hierarchy process: planning, priority setting, resource allocation*. RWS, Pittsburgh, 2nd edition, 1990.

[5] Georg Herzwurm, et al. QFD for Customer-Focused Requirements Engineering. In *Proc. of RE03*, pp. 330–338, Sep. 2003.

[6] International Standard ISO/IEC 9126-1. Software engineering - Product quality - Part 1: Quality model, 2001.

[7] Lawrence Chung, et al. *Non-functional Requirements in Software Engineering*. Kluwer, 2000.

[8] Kyo C. Kang, et al. Peterson. Feature-Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21 ESD-90-TR-222, CMU, Nov. 1990.

[9] Julio Leite et al. A Strategy for Conceptual Model Acquisition. In *Proc. of RE93*, pp. 243–246, 1993.

[10] Steven L. Cornford, et al. Design and Development Assessment. In *Proc. of IWSSD'00*, pp. 105–114, 2000.

[11] Barry Boehm, et al. Developing Groupware for Requirements Negotiation: Lessons Learned. *IEEE Software*, Vol. 18, No. 3, pp. 46–55, May/Jun. 2001.